Counter Machines and Counter Languages[*,†]

by

PATRICK C. FISCHER[‡]
Cornell University
Ithaca, New York

and

ALBERT R. MEYER[¶] and ARNOLD L. ROSENBERG
IBM Watson Research Center
Yorktown Heights, New York

ABSTRACT

The languages recognizable by time- and space-restricted multiple-counter machines are compared to the languages recognizable by similarly restricted multiple-tape Turing machines. Special emphasis is placed on languages definable by machines which operate in "real time". Time and space requirements for counter machines and Turing machines are analyzed. A number of questions which remain open for time-restricted Turing machines are settled for their counter machine counterparts.

Introduction

In recent years there has been increasing interest in analyzing the complexity of Turing machine computations. Paralleling the work on time- and space-restricted Turing machines, we consider similar restrictions on machines with counters in place of tapes. Although counter machines (CM's) appear to be much weaker than Turing machines (TM's), unrestricted CM's have been shown by Minsky [8] to be as powerful as TM's. Restricted counter machines are somewhat more tractable than are restricted Turing machines; therefore, a number of questions which remain open about restricted TM's are settled in this paper for restricted CM's.

In this paper we view machines principally as language recognizers. Many of the properties of TM recognizers carry over to CM's: e.g., the

*Preparation of this paper was partially supported by National Science Foundation Grant GP-7701.

†A portion of this paper was presented at the Eighth Annual Symposium on Switching and Automata Theory, Austin, Texas, October, 1967.

‡Currently at University of British Columbia, Vancouver, B. C., Canada.

¶Currently at Carnegie-Mellon University, Pittsburgh, Pennsylvania.

MATHEMATICAL SYSTEMS THEORY, Vol. 2, No. 3
Published by Springer-Verlag New York Inc.

closure properties of the classes of real-time recognizable languages. On the other hand, a number of significant differences between TM's and CM's can be demonstrated. Throughout this paper we shall contrast corresponding properties of the two models.

Section 1 contains a formal definition of a counter machine recognizer. Some results are proved concerning the relation of the defined model to certain variants of the model. In Section 2 we consider some specific recognition problems in order to lend some insight into the computational capabilities of CM's. Section 3 is concerned with space requirements of counter machines. We establish a straightforward relation between time and space requirements for CM's, and we demonstrate an isomorphism between the hierarchies arising from space bounds on CM's and on TM's. Section 4 contains some remarks about the time required to simulate one kind of machine on another. Finally, in Section 5, we investigate a number of properties of the classes of languages which are definable by CM's with various numbers of counters. We show these classes to form a hierarchy under set inclusion, and we establish a number of their closure properties.

1. The Model and Some Variants

1.1. Definitions

An *alphabet* Σ is a finite set of elements called letters. A *word* over Σ is any finite (possibly null) sequence of letters of Σ. We let Σ^* denote the set of all words over Σ (including the null word λ), and we call any subset of Σ^* a *language* (over Σ). We refer the reader to Ginsburg [4] for definitions of some of the standard operations on words and languages.

A *one-way* (on-line) *k-counter machine* (k-CM) consists of a finite state control unit, k *counters*, each capable of containing any integer, and an input terminal. The states of the finite control are partitioned into *polling* and *autonomous* states. At the start of a computation the CM is in a designated initial state and all counters are set to zero. A *step* in a CM computation is uniquely determined by the state of the control unit, by the symbol scanned at the input terminal if the state is a polling state, and by the set of counters which contain zero. The action at that step consists of independently altering the contents of each counter by adding $0, +1$ or -1 and changing the state of the control unit.

More precisely, a k-CM is specified by

(1) a finite nonempty set Q_p of *polling states*;
(2) a finite set Q_a of *autonomous states*;
(3) a finite *input alphabet* Σ;
(4) a *state transition function*

$$M: (Q_a \cup (Q_p \times \Sigma)) \times \{0, 1\}^k \rightarrow Q_a \cup Q_p;$$

(5) a *counter updating function*

$$K: (Q_a \cup (Q_p \times \Sigma)) \times \{0, 1\}^k \rightarrow \{-1, 0, 1\}^k;$$

(6) a designated *initial state* s_0 in Q_p;
(7) a designated subset F of Q_p, the *final* states.

Let Z denote the set of all integers and Z^k the set of all k-tuples of integers. For any integer x, define the function

$$\mathrm{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x \neq 0, \end{cases}$$

and extend sg to Z^k by

$$\mathrm{sg}(\langle x_1, \cdots, x_n \rangle) = \langle \mathrm{sg}(x_1), \cdots, \mathrm{sg}(x_k) \rangle .$$

A *configuration* of a k-CM is a member of $(Q_a \cup Q_p) \times \Sigma^* \times Z^k$. We write

$$(q, aw, \langle x_1, \cdots, x_k \rangle) \rightarrow (q'', w, \langle y_1, \cdots, y_k \rangle),$$

or

$$(q', w, \langle x_1, \cdots, x_k \rangle) \rightarrow (q'', w, \langle z_1, \cdots, z_k \rangle),$$

for q in Q_p, q' in Q_a, a in Σ, w in Σ^*, and the x_i, y_i, and z_i in Z if

(1) $$M(q, a, \mathrm{sg}(\langle x_1, \cdots, x_k \rangle)) = q''$$

and

$$\langle x_1, \cdots, x_k \rangle + K(q, a, \mathrm{sg}(\langle x_1, \cdots, x_k \rangle)) = \langle y_1, \cdots, y_k \rangle,$$

or

(2) $$M(q', \mathrm{sg}(\langle x_1, \cdots, x_k \rangle)) = q''$$

and

$$\langle x_1, \cdots, x_k \rangle + K(q', \mathrm{sg}(\langle x_1, \cdots, x_k \rangle)) = \langle z_1, \cdots, z_k \rangle .$$

For configurations C and C' we write $C \Rightarrow C'$ if either $C = C'$ or if there exist configurations $C = C_0, C_1, \cdots, C_t = C'$ such that $C_i \rightarrow C_{i+1}$ for each $0 \leq i < t$.

A k-CM *halts in t steps* in its computation of a word w if there are configurations $C_0, C_1, \cdots, C_t$ of the k-CM with

$$(s_0, w, \langle 0, \cdots, 0 \rangle) = C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \cdots \rightarrow C_t = (q, \lambda, \langle x_1, \cdots, x_n \rangle)$$

with q in Q_p. The CM *accepts* or *rejects* w according as q is in F or $Q_p - F$, respectively. The *space* required by the CM in processing w is the sum of the maximum absolute values of the contents of each counter in the course of the t steps of the computation.

Let T and S be monotone increasing functions. A counter machine *operates in time T* [*in space S*] if, for all $n \geq 0$, when the CM is started with input w of length n, it halts in $T(n)$ or fewer steps [requires space not exceeding $S(n)$].

A language L over Σ is *recognized* by a counter machine if the CM accepts every word in L and rejects every word in $\Sigma^* - L$. L is *recognizable in time*

$T(n)$ [*in space* $S(n)$], abbreviated $T(n)$-recognizable [$S(n)$-recognizable], if there is a CM recognizing L which operates in time T [in space S].

We single out the case $T(n) = n$ as a *real time.* A language which is recognizable in time $T(n) = n$ is called *real-time recognizable.* The significance of the real-time restriction stems from the importance of finding efficient algorithms for syntactic analysis.

In analogy to our definitions of counter machines and the various time and space restrictions thereon, one can define the corresponding notions for multitape Turing machines. For brevity, we refer the reader to the following papers for the appropriate definitions and related results: Rabin [11], Hartmanis and Stearns [6], Stearns, Hartmanis and Lewis [14], and Rosenberg [13].

1.2. Variants of the Model

Counter machines have been defined in various ways in the literature. It is worthwhile to verify, for the time-restricted case, that our model is not sensitive to mild changes in convention. Two machines will be said to be *time-equivalent* if they both recognize the same language and whenever one operates in time $T(n)$, the other does also.

THEOREM 1.1. *Given any k-CM with the ability to alter the contents of each counter independently by any integer between $+c$ and $-c$ in a single step (for some fixed integer c), one can effectively find a time-equivalent (ordinary) k-CM.*

Proof. We sketch the construction and leave the details to the reader. When the original k-CM has counters containing the integers $m_1, \cdots, m_k$, the "compressed" k-CM has counters containing $[m_1/c], \cdots, [m_k/c]$, retaining the (bounded) residues $m_1 - c[m_1/c], \cdots, m_k - c[m_k/c]$ in finite state memory.[1] Clearly, the counters of the compressed machine need be altered by at most one per step.

THEOREM 1.2. *Given any k-CM, one can effectively find a time-equivalent k-CM which (a) stores only non-negative integers in its counters; (b) alters its counters at most every c step, for some integer $c > 0$; (c) alters at most one counter per step.*

Proof. For part (a) the CM stores the magnitudes of the desired numbers in its counters and retains the signs of the numbers by enlarging the size of the finite-state control unit (by a factor of 2^k). For parts (b) and (c) the compression technique of Theorem 1.1 is used.

Henceforth, we shall generally make the tacit assumption that counters contain only non-negative integers, i.e., that the CM's satisfy part (a) above.

In contrast to these invariance results, we find that allowing a CM to have an input tape on which it has bilateral motion changes minimum time requirements quite drastically. This *two-way* (off-line) model has an input channel on which resides a read-only scanning device. An input word w is encoded on the input channel in the form $\$w\$$ and the read head is initially

[1] $[x]$ denotes the integer part of the real number x.

placed on the leftmost symbol of w (or on the right-hand \$ if $w = \lambda$). At each step, a two-way CM can shift its read head one square left or right on the input as long as it does not leave the region on which the input word is encoded. Accepting and rejecting states are considered *halting* states, and a word is accepted or rejected according to the state in which the CM halts. We refer the reader to Stearns, Hartmanis and Lewis [14] for a lengthier discussion of off-line machines.

One easily verifies that (1) the distinction between one-way and two-way is vacuous in the real-time case; and (2) given any one-way machine, one can effectively find a time-equivalent two-way machine.

To contrast one-way and two-way machines, let us consider the language $L = \{x\ a\ x^T \mid x \in \{0, 1\}^*, a \notin \{0, 1\}\}$, where x^T denotes the reversal of word x.

THEOREM 1.3. *(a) The language L is not recognizable by any one-way k-CM which operates in time less than $T(n) = 2^{n/2k}$. (b) L is recognizable by a two-way CM which operates in time $T(n) = cn^2/[\log n]$ for some constant c. (c) Every CM recognizing L operates in time $T(n) \geq cn^2/[\log n]$ for some constant c and for all n.*

Proof. (a) A *memory configuration* of a one-way k-CM is the $(k + 1)$-tuple consisting of its current internal state and the contents of its k counters. Any one-way k-CM recognizing L must attain distinct memory configurations after reading distinct binary words x and y; otherwise $x\ a\ x^T$ and $y\ a\ x^T$ would be treated identically. Let t be the maximum contents of any counter after reading a length m word. If the k-CM has q states, then the number of distinct memory configurations of the k-CM after reading length m words cannot exceed $q \cdot (t + 1)^k$. By our preceding remarks, we must have $q \cdot (t + 1)^k \geq 2^m$, whence $t \geq c \cdot 2^{m/k}$ for some constant c. Since a counter machine can alter the contents of a counter by at most one at each step, the k-CM must take no fewer than $c \cdot 2^{(n-1)/2k} \geq d \cdot 2^{n/2k}$ steps for some d, to process input words $x\ a\ x^T$ of length n. Hence the result.

(b) We sketch the action of a two-way CM M which recognizes L.

(i) M traverses the input word $x\ a\ y$, storing $[\log m]$ in a counter, where m is the length of x. (All logarithms in this paper are to the base 2.)

(ii) M successively picks up $[\log m]$ binary symbols of x, interpreting them as a dyadic integer (where the dyadic integer represented by the binary string $b_n b_{n-1} \cdots b_0$ will be $\Sigma_{i=0}^{n} (b_i + 1)2^i$), crosses to the corresponding $[\log m]$ symbols of y and checks that these symbols represent the same integer. The modification of this step when the "a" is encountered is obvious.

Now step (i) clearly requires m steps. In step (ii), each number conversion can be done in time $d \cdot 2^{[\log m]} \leq d \cdot m$ steps for some constant d. Similarly, each positioning of the read head can be done in time proportional to m. Thus, each execution of step (ii) can be done in $e \cdot m$ steps for some constant e. Step (i) is executed once, and step (ii) is executed at most

$m/[\log m] + 1$ times. Thus, M recognizes L in time

$$T(n) \leqslant m + e \cdot m(m/[\log m] + 1) \leqslant c \cdot n^2/[\log n]$$

for some constant c, since $n = 2m + 1$.

(c) To show that M operates within a constant factor of best possible time, we paraphrase a result of Cobham [1].

LEMMA. *For any CM which recognizes the language L, we have*

$$\inf_{n\to\infty} \frac{\log S(n)}{\log n} > 0; \tag{1}$$

$$\inf_{n\to\infty} \frac{T(n) \cdot \log S(n)}{n^2} > 0. \tag{2}$$

In Section 3 we prove that (1) implies that $S(n)^c > T(n)$ for some positive constant c. Using this result, (2) reduces to

$$T(n) \cdot (\log T(n)) \geqslant d \cdot n^2 \tag{$*$}$$

for some constant d and for all n. This implies that

$$T(n) \geqslant \frac{d}{2} \frac{n^2}{\log n}$$

for all n, as the reader may verify.

This establishes part (c), completing the proof.

2. Special Recognition Problems

The purpose of this section is to lend insight into the capabilities of counter machines. It is tautologous to remark that counter machines can solve those problems which can be couched in terms of counting. It is, however, rather surprising to note the breadth of the problems that can be so approached. In fact, Minsky [8] has proved that, in the absence of time restriction, any (partially) computable problem can be rephrased as a counting problem.

We now consider three problems which can be solved in real-time counter machines. No proofs will be given in this section; references to appropriate sources will appear instead.

2.1. Arithmetic Expressions

The first language we consider is the set of all well-formed arithmetic expressions in parenthesis-free notation. Assuming only unary and binary operators, the language L_2 (the subscript denoting the highest ranking operator) is schematically defined by the grammar:

⟨expression⟩ → ⟨unary operator⟩ ⟨expression⟩
⟨expression⟩ → ⟨binary operator⟩ ⟨expression⟩ ⟨expression⟩
⟨expression⟩ → ⟨operand⟩.

The reader can easily supply the general definition of L_n.

THEOREM 2.1. *For all n, the language L_n is real-time recognizable by a 1-CM.*

Proof. The proof follows from an algorithm presented by Oettinger [9]. One assigns a numerical value to each symbol in the vocabulary: -1 to each operand, and $+(n-1)$ to each n-ary operator ($n \geq 1$). The 1-CM reads the input word, responding to each symbol by adding its numerical value to the counter. The input word is accepted if, and only if, (i) at the end of the word the counter contains -1, and (ii) at no previous point in the computation did the counter contain a negative number. One readily verifies that the CM recognizes L_n. The theorem now follows by Theorem 1.1.

Thus, even real-time one-counter machines can perform some relatively sophisticated computations.

2.2. Commutative Languages

A language L is *commutative* if all permutations of every word in L are also in L. Clearly, in recognizing a commutative language, one need only consider the number of occurrences of the various vocabulary symbols in the input word.

Given a vocabulary $\Sigma = \{a_1, \cdots, a_n\}$, we define, for any word w over Σ, $\#(w)$ to be the n-tuple of non-negative integers $\langle \#_1(w), \cdots, \#_n(w) \rangle$, where $\#_i(w)$, $1 \leq i \leq n$, is the number of occurrences of a_i in w.

A set S of n-tuples of integers is *linear* if there exist non-negative n-tuples $c, p_1, \cdots, p_r$ such that $S = \{c + \Sigma_{i=1}^{r} k_i p_i |$ each $k_i \geq 0\}$. S is *semilinear* if it is a finite union of linear sets.

Using results of Ginsburg and Spanier [5] and an application of Cramer's rule for solving systems of linear equations, one can show that for every semilinear set of n-tuples of integers S, and for every n-letter vocabulary $\Sigma = \{a_1, \cdots, a_n\}$, the language $\{w \in \Sigma^* | \#(w) \in S\}$ is equal to a finite Boolean combination of sets which are real-time recognizable by 1-CM's. Conversely, since a real-time 1-CM obviously accepts only context-free languages (cf. Fischer [2]), Parikh's theorem [10] shows that, for any language L real-time recognizable by a 1-CM, the set $\{\#(w) | w \in L\}$ is semilinear. We thus obtain the following result which was obtained in slightly different form by Laing [7].

THEOREM 2.2. (Laing [7]) *Let S be a set of n-tuples of integers, and let $\Sigma = \{a_1, \cdots, a_n\}$. The language L over Σ equal to $\{w \in \Sigma^* | \#(w) \in S\}$ is a finite Boolean combination of languages real-time recognizable by 1-CM's if, and only if, S is a semilinear set.*

Two examples of commutative languages are of special interest and are worthy of mention.

2.3. Walks in *n*-Space

It is trivial to show that, given any real-time n-CM, one can find a time-equivalent n-tape TM. In view of the greater flexibility of TM tapes, it is natural to wonder if a real-time TM with fewer than n tapes can be found to simulate a real-time n-CM. We present one instance where this question is still open, and mention one for which the question has recently been resolved in the affirmative.

Let $\Sigma_n = \{a_1, a_2, \cdots, a_{2n-1}, a_{2n}\}$ be an alphabet of $2n$ letters.

Axis-Crossing Problem. The n-dimensional axis-crossing problem is the language

$$A_n = \{w \in \Sigma^* | \text{ for some } i,\ 1 \leqslant i \leqslant n,\ \#_{2i-1}(w) = \#_{2i}(w)\}.$$

Open Problem. It is obvious that, for all n, A_n is real-time recognizable by an n-CM. Is A_2 real-time recognizable by a 1-TM?

Origin-Crossing Problem. The n-dimensional origin-crossing problem is the language

$$O_n = \{w \in \Sigma_n^* | \text{ for each } i,\ 1 \leqslant i \leqslant n,\ \#_{2i-1}(w) = \#_{2i}(w)\}.$$

The geometric names of these languages arise from the interpretation of each letter $a_{2i-1}[a_{2i}]$ as an instruction to take one step left [right] along the ith axis of n-space.

In contrast to the above open problem about A_n, for O_n we have the following result of M. Fischer and A. Rosenberg [3].

THEOREM 2.3. *For all $n \geqslant 1$, the language O_n is real-time recognizable by a 1-tape Turing machine.*

COROLLARY. *Let $\Sigma = \{a_1, \cdots, a_n\}$. The languages $\{x\beta y | x, y \in \Sigma^*, \beta \notin \Sigma$, and x is a permutation of $y\}$ and $\{w \in \Sigma^* | \#_1(w) = \#_2(w) = \cdots = \#_n(w)\}$ are real-time recognizable by 1-TM's.*

3. Space Requirements

3.1. Space Complexity Classes

Stearns, Hartmanis, and Lewis [14] have defined, for a function S, the *tape complexity class* C_S to be the class of languages recognizable by TM's which operate in space S. Tape complexity classes, partially ordered by set inclusion, have a rich structure which includes infinitely many incomparable infinite chains. If one similarly defines C_S' to be the class of languages recognizable by CM's which operate in space S, then one obtains the same structure. In this section we investigate the classes C_S'. The main result of the section is that $C_S' = C_{\log(S)}$ for any function S.

We first exhibit two space compression results for CM's. The fact that these compressions can be effected with *no time loss* will be useful in later sections.

LEMMA 3.1. *Given any k-CM which operates in space $S(n)$, one can find a time-equivalent k-CM which operates in space $[c \cdot S(n)]$ for any constant $c > 0$.*

The Lemma follows from Theorem 1.2(b).

At the cost of adding more counters to a CM, Lemma 3.1 can be significantly improved.

LEMMA 3.2. *Let S be an increasing function. Given any CM which operates in space bounded by a polynomial in S, one can find a time-equivalent CM which operates in space S.*

Proof. We describe how any counter of a CM can be replaced by two counters whose contents remain bounded by the square root of the contents of the original counter. By applying this construction several times, the result will follow.

The simulation proceeds in two alternating modes, the current mode being remembered in finite memory. Mode I corresponds to the case where the counter to be simulated contains an integer $m = r^2 + 2i$ $(0 \leqslant i \leqslant r)$, mode II to the case where the counter contains $m = r^2 + 2i + 1$ $(0 \leqslant i < r)$.

Mode I: Suppose that the counter to be simulated contains $m = r^2 + 2i$. We replace that counter by two counters A and B containing $r - i$ and i respectively. To simulate the incrementing of the original counter by +1 we distinguish two cases. If $i = r$, i.e., counter A contains 0, then counter B is incremented by +1, the roles of counters A and B are interchanged, and the machine remains in mode I. If $i \neq r$, the counters remain unchanged, but mode II is entered.

Mode II: If the counter to be simulated contains $m = r^2 + 2i + 1$, then once again, counter A contains $r - i$ and counter B contains i. To simulate the incrementing of the original counter by +1, counter A is decreased by 1, counter B is increased by 1 and mode I is entered.

Simulated decrementing of the original counter proceeds by reversing the process of simulated incrementing with the obvious minor modifications. Clearly the original counter contains zero when, and only when, both counters A and B contain zero. Similarly, when the original counter contains an integer $m \geqslant 0$, one of counters A and B contains $i = [(m - [\sqrt{m}]^2)/2] \leqslant [\sqrt{m}]$, and the other contains $[\sqrt{m}] - i \leqslant [\sqrt{m}]$. Finally, our new CM requires only a single step to simulate a step of the original counter.

Using Lemmas 3.1 and 3.2, we can prove the main result of this section, namely that the TM and CM space hierarchies are isomorphic.

THEOREM 3.1. *A language is recognizable by a CM which operates in space S if, and only if, it is recognizable by a TM which operates in space* log (S).

Proof. By encoding the contents of the various counters of the CM on separate tracks of its tape in (possibly compressed) binary notation, a TM can clearly simulate a CM which operates in space S, using space bounded by log (S).

For the converse we employ an algorithm reported in Fischer [2] for simulating a Turing machine tape using three counters, A, B and C.

Assume that the TM employs m distinct symbols, labelled 0 (blank), $1, \cdots, m - 1$. If at some point the nonblank portion of the TM tape is of the form $a_r a_{r-1} \cdots a_0 c b_0 b_1 \cdots b_s$ with the read-write head residing on the c, then at the corresponding point in the simulation, counter A will contain the integer

$$|a_r| \cdot m^r + |a_{r-1}| \cdot m^{r-1} + \cdots + |a_1| \cdot m + |a_0|,$$

counter B will contain

$$|b_s| \cdot m^s + |b_{s-1}| \cdot m^{s-1} + \cdots + |b_1| \cdot m + |b_0|,$$

and counter C will contain zero. (We use the notation $|a_k|$ to denote the integer label of symbol a_k.) The integer $|c|$ is retained in finite memory.

If the TM replaces c by d and shifts its head right, the CM multiplies the contents of counter A by m and adds $|d|$, and it divides the contents of counter B by m, retaining the residue $|b_0|$ in finite memory. Left shifts are simulated analogously. Throughout this simulation, counter C is used as an auxiliary register.

It is now clear that, if the TM scans at most $[\log S]$ squares of tape, then counters A, B and C never grow larger than

$$m^{[\log S]+1} \leqslant mS^{[\log m]+1} .$$

The result now follows by Lemma 3.2.

3.2. Time-Space Relation

There is a strong relation between time and space requirements for counter machines. This relation is made explicit in the following theorem.

THEOREM 3.2. *Let S be a function with $S(n) \geqslant n$. A language L is recognizable by a CM which operates in space S if, and only if, it is recognizable by a CM which operates in time bounded by a polynomial in S.*

Proof. Let L be recognized by a CM with q states and k counters which operates in space S. It follows that, in processing an input of length n, the CM can assume the most $q \cdot n \cdot (S(n) + 1)^k$ distinct configurations. (See Section 1.1 for the definition of the *configuration* of a k-CM.) Clearly, if a configuration is repeated in the course of a computation, the CM will never halt. Therefore, since by definition of recognition the CM always halts, the CM must operate in time

$$T(n) = q \cdot n \cdot (S(n) + 1)^k \leqslant c \cdot S(n)^{k+1}$$

for some constant c independent of n.

Conversely, if time is bounded by a polynomial in S, then so is space since a CM can increment its counters by at most one at each step. We now appeal to Lemma 3.2 to complete the proof.

Open Problem. Is there an analogue of Theorem 3.2 which is valid for Turing machines?

3.3. Basis for Complexity Classes

Stearns, Hartmanis and Lewis [14] remark that to study any space complexity class C_S of TM's, it suffices to consider one-tape TM's. We now show this not to be true for the CM complexity classes. In particular we show that the class of languages recognizable by $(k+1)$-CM's which operate in space $S(n) = n$ properly includes the class recognizable by k-CM's which operate in space $S(n) = n$ for any integer k.

For $k = 1, 2, \cdots$, let L_k be the language $L_k = \{0^{m_1}10^{m_2}1 \cdots 0^{m_k}\beta_i 0^{m_i} | 1 \leqslant i \leqslant k$, each $m_j \geqslant 1$, and each $\beta_i \notin \{0, 1\}\}$.

THEOREM 3.3. *(a) For each k, the language L_k is recognizable by a (real-time) k-CM which operates in space $S(n) = n$. (b) If an r-CM M recognizes L_k, for $k \geqslant r$, then M operates in space $S(n) \geqslant cn^{k/r}$ for some constant c.*

Proof. The proof of part (a) is straightforward and is left to the reader. To establish (b), note that distinct binary words x of the form $0^{m_1}1 \cdots 10^{m_k}$ must leave M in distinct memory configurations. Now if each $m_i \leqslant h$, the number of such distinct words is h^k. If t is the maximum contents attained by any counter of M in processing such words, then the number of distinct memory configurations of M is no greater than $q(t+1)^r$, where q is the number of states of M. We must then have $q(t+1)^r \geqslant h^k$, whence $t \geqslant d \cdot h^{k/r}$ for some constant d. Since $h + 1 \geqslant n/(k+1)$, where n is the length of the input word, M must operate in space $S(n) \geqslant c \cdot n^{k/r}$ for some constant c, completing the proof.

COROLLARY. *For all k, the class of languages recognized by $(k+1)$-CM's in space $S(n) = n$ properly includes the corresponding class recognized by k-CM's.*

4. Simulation of One Machine by Another

In this section we investigate precise time bounds for one machine to simulate another. For simplicity we restrict attention to the problem of simulating machines which operate in *linear time,* that is, in time $T(n) = cn$ for some constant c. The generalization of several of the results to arbitrary $T(n)$ is immediate.

4.1. Turing Machines and Counter Machines

It is obvious that, for any timing function $T(n)$, given any k-CM which operates in time $T(n)$, one can find a time-equivalent k-TM. It is, however, rather surprising to note that a 1-TM can simulate a real-time k-CM with little time loss and can simulate a k-CM which operates in time $T(n) > (1+\epsilon) \cdot n$ with no time loss.

THEOREM 4.1. *Given any k-CM which operates in (linear) time $T(n) \leqslant c \cdot n$ for any $\epsilon > 0$, one can effectively find an equivalent one-tape TM which operates in time $T(n) = (1+\epsilon)n$.*

Proof. The proof proceeds in three stages. (a) Given a k-CM M, we find a time-equivalent k-CM M' which alters at most one counter per step. (b) Next we construct a single-tape TM T, equivalent to M', which operates in time $T(n) = 12cn$. (c) Finally we obtain, for any desired real number $d > 0$, a single-tape TM T', equivalent to T, which operates in time $T(n) = (1 + (12c - 1)d)n$. Since the feasibility of steps (a) and (c) are demonstrated by Theorem 1.2 and by the speed-up theorem of Hartmanis and Stearns [6], we consider only step (b).

Given M', we construct T as follows. The tape of T is divided into k *tracks,* one for each counter of M', and each track is divided into two *channels.* A binary representation of a non-negative integer will appear in each track, justified so that the low-order bits of all integers appear in the same tape square. This encoding is such that, for $i = 1, \cdots, k$, if the ith counter of M' contains, at the nth step in the computation, the integer x_i, then at the nth stage in the simulation, the ith track of the tape of T will contain integers y_i and z_i with the properties: (1) $y_i - z_i = x_i$, (2) $y_i + z_i \leq n$, and (3) there is no bit position in which the binary representations of y_i and z_i both contain a one. This last condition assures us that $x_i = 0$ when, and only when, $y_i = z_i = 0$.

For the case $k = 3$, Figure 1 illustrates a possible configuration of the tape of T after 15 steps by M'. The counters of M' contain 7, −5 and 0, respectively.

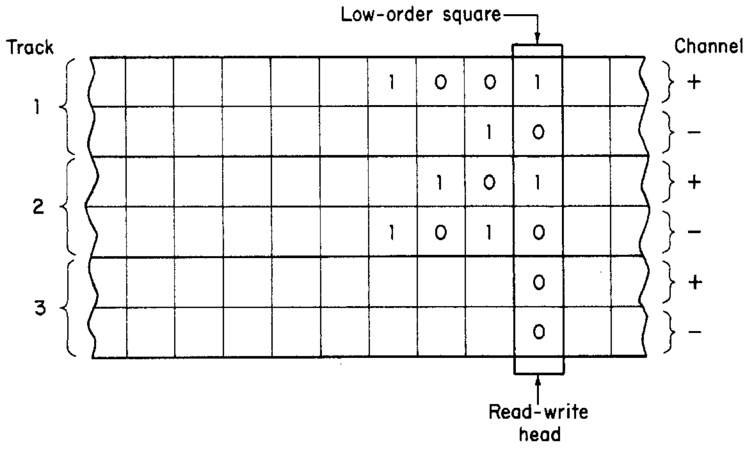

Figure 1. A possible configuration of T's tape after 15 steps by M': Counter 1 contains 7, counter 2 contains −5, and counter 3 contains 0.

The process of simulating a step of M' proceeds as follows: If counter i is to be incremented [decremented], 1 is added to y_i [z_i]. After any necessary carries are performed, T moves one additional square to the left in order to determine if the leading digit of either channel of track i has been reached. T then performs a "clean-up" while shifting right until the blank square to the right of the low-order square is detected. In this clean-up, T maintains condition (3) above and replaces any leading zeros thus created by blanks. Finally, T shifts left one square to return to the low-order position. It is clear that during the clean-up, T also detects whether the value of counter i is now zero or nonzero.

Figure 2 illustrates the change in the configuration of the tape of T which results from adding one to the second counter of M' when M' is in the configuration of Figure 1.

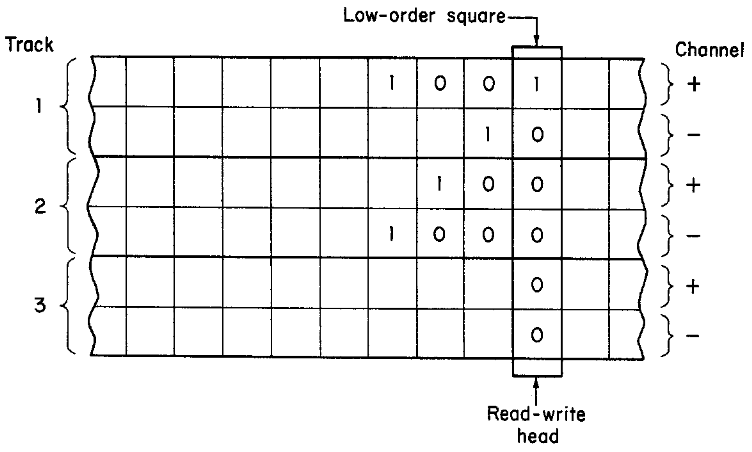

Figure 2. The configuration of T's tape after simulating the incrementing of counter 2 of M' by 1.

To estimate the time required for a complete simulation of M' by T we note that in the simulation of the first 2^{h+1} steps of the computation of M', the worst case for T will occur when each step increases the same counter of M'. In this case, there will be 2^h steps in which no carry is required, 2^{h-1} steps involving a carry of 1, 2^{h-2} steps involving a carry of 2, etc., 1 step with a carry of h and 1 step with a carry of $h + 1$. Since the simulation of a step of M' involving a carry of length i takes $2i + 4$ steps by T, we see that the time taken by T is:

$$\begin{aligned}
T(2^{h+1}) &= \sum_{i=0}^{h} 2^{h-i}(2i + 4) + 2(h + 1) + 4 \\
&= \sum_{i=0}^{h} 2^{h-i}(2i + 4) + \sum_{i=h+1}^{\infty} 2^{h-i}(2(h + 1) + 4) \\
&\leqslant \sum_{i=0}^{\infty} 2^{h-i}(2i + 4) \\
&= 2^h(2 \sum_{i=0}^{\infty} i2^{-i} + 4 \sum_{i=0}^{\infty} 2^{-i}) \\
&= 2^h \cdot 12
\end{aligned}$$

Now, given n, choose $h = [\log n]$ so that $2^h \leqslant n < 2^{h+1}$. Then $T(n) \leqslant T(2^{h+1}) \leqslant 12 \cdot 2^h \leqslant 12n$. (A careful analysis would show that a bound of $6n$ actually holds.) Thus, Theorem 4.1 follows.

Open Problem. Can the ϵ of Theorem 4.1 be set to 0? This problem is open even for the case of real-time 2-CM's. (Cf. the axis-crossing problem in Section 2.2.)

In contrast to Theorem 4.1, a one-way CM may require exponential time to simulate a real-time one-tape TM.

THEOREM 4.2. *(a) Given any linear-time k-tape TM, one can find an equivalent one-way CM which operates in time $T(n) = a^n$ for some constant a. (b) There is a language L, real-time recognizable by a one-tape TM, which is not recognizable by any k-CM which operates in time less than $T(n) = 2^{n/2k}$.*

Proof. (a) A linear-time TM operates in space $S(n) = cn$. Part (a) thus follows from Theorems 3.1 and 3.2.

(b) The language $L = \{x\, a\, x^T \mid x \in \{0, 1\}^*, a \notin \{0, 1\}\}$ is easily shown to be real-time recognizable by a one-tape TM (in fact by a real-time pushdown automaton). Therefore, part (b) follows from the proof of Theorem 1.3(a).

We can thus bound the time required for a CM to simulate a linear-time TM by the inequalities

$$c_1^n \leq T(n) \leq c_2^n$$

for some constants c_1 and c_2.

4.2. *k*-Counter Machines and 3-Counter Machines

THEOREM 4.3. *(a) Given any linear-time k-CM, one can find an equivalent 3-CM which operates in time $T(n) = cn^{k+2}$ for some constant c. (b) There is a language L, real-time recognizable by a k-CM, which is not recognizable by any r-CM which operates in time less than $T(n)^{k/r}$.*

Proof. (a) Given a k-CM M which operates in time $d \cdot n$, for some constant d, one easily constructs an equivalent 1-TM which (i) operates in time $d'n \cdot \log n$ (for some d'), (ii) operates in space $\log dn$, and (iii) uses $2^k + 1$ working symbols. The 1-TM encodes the contents of the counters of M in binary on its tape using *one* track of tape per counter. It simulates an alteration to the counters of M by altering the binary integers on the corresponding tracks of its tape. One easily verifies that, if we assume that M alters at most one counter at each step, then a single step of M, while M is checking the nth input symbol, requires no more than $\log dn$ steps of the 1-TM. Thus the 1-TM satisfies the assertions above.

We now employ the algorithm of Theorem 3.1 to find a 3-CM to simulate the 1-TM just constructed. Now to simulate a single step of the 1-TM which, in turn, is working on the simulation of M checking the nth input symbol, the 3-CM must multiply a number of magnitude at most $(2^k + 1)^{\log dn} \leq e \cdot n^{k+1/2}$ (for some constant e) by $2^k + 1$, divide a number of similar magnitude by $2^k + 1$, and add a number of magnitude at most $2^k + 1$. Each such simulated step thus requires at most $3 \cdot (2^k + 1) \cdot e \cdot n^{k+1/2}$ steps of the 3-CM.

Since the 1-TM operates in time $d' \cdot n \cdot \log n$, the simulating 3-CM operates in time $h \cdot n^{k+3/2} \cdot \log n \leq c \cdot n^{k+2}$, as was asserted.

(b) We now recall the family of languages

$L_k = \{0^{m_1}10^{m_2}1 \cdots 0^{m_k}\beta_i 0^{m_i} \mid 1 \leq i \leq k$, each $m_j \geq 1$, and each $\beta_i \notin \{0, 1\}\}$.

It is obvious that each L_k is real-time recognizable by a k-CM. It follows from the proof of Theorem 3.3(b) that no r-CM recognizing L_k operates in time less than $cn^{k/r}$ for some constant c. The theorem follows.

5. Real-Time Recognizable Languages

Let $\mathscr{C}_k$ denote the class of languages real-time recognizable by k-CM's ($k = 0, 1, 2, \cdots$), and let $\mathscr{C} = \bigcup_k \mathscr{C}_k$. In this section we investigate a number of properties of the classes $\mathscr{C}_k$ and of $\mathscr{C}$.

5.1. The Counter Hierarchy

Most of the negative results in this section depend on the following basic lemma.

We say that two words x and y are *n-equivalent with respect to a language L*, denoted $x\, E_n\, y \pmod{L}$, if, for all words z of length not exceeding n, xz is in L when, and only when, yz is in L.

Let $T(M)$ denote the language recognized by the CM M.

LEMMA 5.1. *Let M be a real-time k-CM with q states. The number of equivalence classes of* $E_n(\text{mod } T(M))$ *cannot exceed* $q \cdot (n + 1)^k \leq cn^k$ *for some constant c.*

The proof is obvious when one considers the number of distinct memory configurations of M which can be distinguished in n steps (which are *n-inequivalent*).

Lemma 5.1 immediately yields the following theorem.

THEOREM 5.1. *For all k,* $\mathscr{C}_k$ *is properly included in* $\mathscr{C}_{k+1}$.

Proof. Consider the family of languages L_k of Theorems 3.3(b) and 4.3(b). Given any distinct words $x = 0^{m_1}1 \cdots 10^{m_{k+1}}$ and $y = 0^{r_1}1 \cdots 10^{r_{k+1}}$, where each m_i and $r_j \leq h$, there is a sequence $z = \beta_s 0^t$ of length not exceeding $h + 1$ such that xz is in L_{k+1} while yz is not. Since the number of distinct words of this form is h^{k+1}, it follows that the number of equivalence classes of $E_{h+1}(\text{mod } L_{k+1})$ is no less than h^{k+1}.

Now, since the number of equivalence classes of $E_{h+1}(\text{mod } T(M))$ for any real-time k-CM M is no greater than $q(h + 1)^k < h^{k+1}$ for large h, it follows that L_{k+1} is not in $\mathscr{C}_k$.

It is obvious that L_{k+1} is in $\mathscr{C}_{k+1}$, whence for all k, L_{k+1} is in $\mathscr{C}_{k+1} - \mathscr{C}_k$. The theorem follows.

Lemma 5.1 and Theorem 5.1 were noted independently by Laing [7].

5.2. Linear Time and Speed-up

Using part (b) of Theorem 1.2, one immediately obtains a "speed-up" theorem for counter machines.

THEOREM 5.2. *Given a k-CM which operates in time* $T(n) = n + E(n)$,

$E(n) \geqslant 0$, one can find an equivalent k-CM which operates in time $T'(n) = n + [cE(n)]$ for any constant $c > 0$.

In particular, we have the following corollary.

COROLLARY. *Given any k-CM which operates in time $T(n) = d \cdot n$, $d \geqslant 1$, one can find an equivalent k-CM which operates in time $(1 + [(d-1)/c]) \cdot n$ for any constant $c > 0$.*

Thus any linear time CM can be replaced by an equivalent CM which operates in time $(1 + \epsilon) \cdot n$. It is natural to wonder if the ϵ can be set to zero, establishing the equivalence of linear time and real time. We now show this not to be the case.

THEOREM 5.3. *Let $L = \{0^p 1^m | \, p \geqslant m > 0\}$. The language L^* is (a) recognizable in time $2n$ by a 1-CM, but (b) not real-time recognizable by any CM.*

Proof. Part (a) is obvious. We establish part (b) by contradiction.

Assume that the k-CM M, having q states, recognizes L^* in real time. Let $b_0 = q$, and for $i = 1, \cdots, k$, let $b_i = q \cdot (3b_{i-1} + 2)^i$. We shall show that M must accept a string of the form

$$0^{c_k} 1 0^{c_{k-1}} 1 \cdots 0^{c_1} 1 0^d 1^e$$

where $d < e$, or reject such a string where $d \geqslant e$. The integers c_i will satisfy the inequalities

$$1 \leqslant c_i \leqslant b_i \qquad (i = 1, \cdots, k).$$

Now, for any integers p and $m > p$, M must be in distinct memory configurations after reading 0^p and 0^m; otherwise $0^m 1^m \in L^*$ and $0^p 1^m \notin L^*$ would be treated identically. Since the number of distinct memory configurations is bounded by $q \cdot (t+1)^k$, where t is the largest integer attained by any counter, it follows that after reading some word 0^{c_k} for $c_k \leqslant b_k$, some counter of M must contain $t \geqslant 3b_{k-1} + 1$. Therefore, after reading $0^{c_k}1$, some counter of M must contain an integer not less than $3b_{k-1}$.

Assume, for induction, that for $0 < i < k$, integers $c_k, c_{k-1}, \cdots, c_{k-i+1}$ have been chosen so that when M is scanning the final symbol of the word

$$0^{c_k} 1 0^{c_{k-1}} 1 \cdots 1 0^{c_{k-i+1}} 1$$

at least i of the counters of M contain integers not less than $3b_{k-i}$. As above, when M now scans input words of the form 0^p and 0^m for $b_{k-i} \geqslant m > p \geqslant 1$, it must enter configurations which are b_{k-i}-inequivalent. By our inductive hypothesis, i of the counters of M are too large to affect b_{k-i}-equivalence of configurations. Therefore, M can now attain no more than $q \cdot (t+1)^{k-i}$ b_{k-i}-inequivalent configurations, where t is the largest integer attained by any of the $k - i$ "unclogged" counters. It follows that there is an integer $c_{k-i} \leqslant b_{k-i}$ such that, after reading the word

$$0^{c_k} 1 \cdots 0^{c_{k-i+1}} 1 0^{c_{k-i}} 1,$$

at least $i + 1$ of the counters of M contain integers not less than $3b_{k-i-1}$.

By induction, there must be an input word $w = 0^{c_k}1 \cdots 0^{c_1}1$, where each $c_i \leq b_i$ $(i = 1, \cdots, k)$ such that, after reading w, all of the counters of M contain integers no less than $3b_0 = 3q$. There must now be integers $1 \leq c < d \leq q + 1$ such that $w0^c$ and $w0^d$ lead M to q-equivalent configurations. Therefore, M must treat $w0^c1^{c+1} \notin L^*$ and $w0^d1^{c+1} \in L^*$ identically, contradicting the assumption that M recognizes L^*. The proof is completed.

Theorem 5.3 thus demonstrates the inequivalence of linear time and real time for CM's.

It is worthwhile to note that the language L^* is real-time recognizable by a 1-CM which can, in a single step, set its counter to zero. Theorem 5.3 thus demonstrates the

Remark. The class of languages real-time recognizable by CM's with "store zero" instructions properly includes $\mathscr{C}$.

The reader will easily verify that almost all the results reported in this paper remain valid for these strengthened CM's.

5.3. Closure Properties

In this final section we investigate the closure properties of the classes $\mathscr{C}_k$ and of $\mathscr{C}$.

The closure properties of the class $\mathscr{C}$ are almost identical to those of the class of real-time definable languages (Rosenberg [13]).

THEOREM 5.4. *The class $\mathscr{C}$ is closed under the operations of (a) complementation, (b) union, (c) intersection, (d) suffixing with a regular set, and (e) inverse generalized sequential machine mapping.*

The constructions required to prove Theorem 5.4 are virtually identical to those used by Rosenberg [13] to establish the analogous results for Turing machines. We refer the reader to Rosenberg's paper for these constructions.

LEMMA 5.2. *The language $L = \{0^{m_1}10^{m_2}1 \cdots 0^{m_{r-1}}10^{m_r}20^m \mid r \geq 1$, each $m_i \geq 1$, and for some $1 \leq i \leq r$, $m = m_i\}$ is not in $\mathscr{C}$.*

Proof. Given any distinct binary words of the form $x = 0^{m_1}1 \cdots 0^{m_r}1$ $(r \geq 1, 1 \leq m_j \leq h)$ and $y = 0^{n_1}1 \cdots 0^{n_s}1$ $(s \geq 1, 1 \leq n_j \leq h)$, then, whenever $\{m_1, \cdots, m_r\} \neq \{n_1, \cdots, n_s\}$, there is a word $z \in \{2\} \cdot \{0\}^*$ of length not exceeding $h + 1$ such that $xz \in L$ while $yz \notin L$. It thus follows that the number of equivalence classes of $E_n(\bmod L)$ no less than 2^{n-1}. The result now follows by Lemma 5.1.

THEOREM 5.5. *The class $\mathscr{C}$ is not closed under the operations of (a) concatenation, even with a regular set, (b) length-preserving homomorphism, (c) Kleene closure, and (d) reversal.*

Proof. (a)–(c) Let H be the language $H = \{0^m10^{n_1}1 \cdots 0^{n_r}20^m \mid r \geq 0$, each $n_i \geq 1$, and $m \geq 1\}$. Clearly H is in $\mathscr{C}_1$. Let R be the regular set $R = (\{0\}\{0\}^*\{1\})^*$. Since $RH = L$, part (a) follows by Lemma 5.2.

Let f be the (length-preserving) homomorphism defined by: $f(0) = 0$, $f(1) = 1, f(2) = 2, f(3) = 1$. Clearly $R\{0\}\{0\}^*\{3\}H$ is in $\mathscr{C}_1$. However, the image of this language under f is L, establishing (b).

By Theorem 5.4, $G = H\{3\} \cup R$ is in $\mathscr{C}$. However, G^* is not in $\mathscr{C}$, or else, using Theorem 5.4, so would be $G^* \cap R\{2\} \cdot \{0\}^* \cdot \{3\} = L \cdot \{3\}$, contradicting Lemma 5.2. This establishes (c).

(d) One easily verifies that the language $K = \{1^n0^m | \; 0 < n \leq m\}^*$ is in $\mathscr{C}_1$. However, the reversal of K is the language L^* of Theorem 5.3 which we know not to be in $\mathscr{C}$.

We finally note

THEOREM 5.6. *For all $m, n > 0$, there is a language H in $\mathscr{C}_m$ and a language K in $\mathscr{C}_n$ such that $H \cup K$ is in $\mathscr{C}_{m+n} - \mathscr{C}_{m+n-1}$.*

Proof. We merely let

$$H = \{0^{a_1}1 \cdots 0^{a_m}10^{b_1}1 \cdots 0^{b_n}\beta_i 0^{a_i} | \; 1 \leq i \leq m, \text{ each } a_i, b_j \geq 1, \text{ and each } \beta_k \notin \{0, 1\}\},$$

$$K = \{0^{c_1}1 \cdots 0^{c_m}10^{d_1}1 \cdots 0^{d_n}\beta_i 0^{d_i} | \; 1 \leq i \leq n, \text{ each } c_i, d_j \geq 1, \text{ and each } \beta_k \notin \{0, 1\}\}.$$

Clearly $H \cup K$ is the language L_{m+n} of Theorem 5.1, and is, therefore, in $\mathscr{C}_{m+n} - \mathscr{C}_{m+n-1}$.

Open Problem. Is there an analogue of Theorem 5.6 for TM's?

REFERENCES

[1] A. COBHAM, Time and memory capacity bounds for machines which recognize squares or palindromes, *IBM Research Report RC-1621*, 1966.
[2] P. C. FISCHER, Turing machines with restricted memory access. *Information and Control* **9** (1966), 364–379.
[3] M. J. FISCHER and A. L. ROSENBERG, Real-time solutions of the origin-crossing problem. *Math. Systems Theory* **2** (1968), 257–263.
[4] S. GINSBURG, *The Mathematical Theory of Context-Free Languages*, McGraw-Hill, New York, 1966.
[5] S. GINSBURG and E. H. SPANIER, Bounded ALGOL-like languages, *Trans. Amer. Math. Soc.* **113** (1964), 333–368.
[6] J. HARTMANIS and R. E. STEARNS, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* **117** (1965), 285–306.
[7] R. LAING, Realization and complexity of commutative events, *Univ. of Mich. Technical Report 03105-48-T*, 1967.
[8] M. MINSKY, Recursive unsolvability of Post's problem of tag and other topics in the theory of Turing machines, *Ann. of Math.* **74** (1961), 437–455.
[9] A. G. OETTINGER, Automatic syntactic analysis and the pushdown store, *Proc. 12th Ann. Sympos. in Appl. Math.*, 1960, pp. 104–129.
[10] R. J. PARIKH, On context-free languages. *J. Assoc. Comput. Mach.* **13** (1966), 570–581.
[11] M. O. RABIN, Real-time computation, *Israel J. Math.* **1** (1963), 203–211.
[12] M. O. RABIN and D. SCOTT, Finite automata and their decision problems, *IBM J. Res. Develop.* **3** (1959), 114–125.

[13] A. L. ROSENBERG, Real-time definable languages, *J. Assoc. Comput. Mach.* **14** (1967), 645–662.

[14] R. E. STEARNS, J. HARTMANIS and P. LEWIS, Hierarchies of memory-limited computation, *Proc. 6th Ann. Sympos. Switching Circuit Theory and Logical Design,* Ann Arbor, Mich., 1966, pp. 179–190.

(Received 7 December 1967)